IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Utility Patent Application

FIELD WEIGHTING IN TEXT DOCUMENT SEARCHING

Inventor(s):

Stephen Robertson

Hugo Zaragoza

Michael Taylor

Stefan Larimore

Mihai Petriuc

CLIENT'S DOCKET NO. MS306779.01

ATTORNEY'S DOCKET NO. MS1-1842US

EV436702995

FIELD WEIGHTING IN TEXT SEARCHING

Technical Field

The invention relates generally to text document searching, and more particularly to field weighting in text document searching.

Background

In a text document search, a user typically enters a query into a search engine. The search engine evaluates the query against a database of indexed documents and returns a ranked list of documents that best satisfy the query. A score, representing a measure of how well the document satisfies the query, is algorithmically generated by the search engine. Commonly-used scoring algorithms rely on splitting the query up into search terms and using statistical information about the occurrence of individual terms in the body of text documents to be searched. The documents are listed in rank order according to their corresponding scores so the user can see the best matching search results at the top of the search results list.

Many such scoring algorithms assume that each document is a single, undifferentiated string of text. The query of search terms is applied to the text string (or more accurately, to the statistics generated from the undifferentiated text string that represents each document). However, documents often have some internal structure (e.g., fields containing titles, section headings, metadata fields, etc.), and reducing such documents to an undifferentiated text string loses any searching benefit provided by such structural information.

Some existing approaches attempt to incorporate the internal structure of documents into a search by generating statistics for individual document fields and generating scores for individual fields. The score for an individual document is then computed as a weighted sum of scores for its fields. However, in such existing approaches, the weights applied to individual fields of different documents do not adequately consider the influence of document length, field lengths, and the possible combinations of term frequencies of different query terms in different fields on the overall score for a given document.

Summary

Implementations described and claimed herein address the foregoing problems by combining statistical information for each term across document fields in a suitably weighted fashion. Both field-specific term frequencies and field lengths may be considered to obtain a field-weighted document weight for each query term. Each field-weighted document weight can then be combined in order to generate a field-weighted score that is responsive to the overall query.

In some implementations, articles of manufacture are provided as computer program products. One implementation of a computer program product provides a computer program storage medium readable by a computer system and encoding a computer program. Another implementation of a computer program product may be provided in a computer data signal embodied in a carrier wave by a computing system and encoding the computer program.

The computer program product encodes a computer program for executing on a computer system a computer process for determining a field-weighted score for a document having multiple fields relative to a query having a plurality of

query terms. A field-weighted term frequency is computed for each query term based on field weights designated for individual fields in the document. A field-weighted document weight is computed for each query term based on the field-weighted term frequency for each field in the document. The field-weight score is computed based on the field-weighted document weights of the query terms.

In another implementation, a method is provided that determines a field-weighted score for a document having multiple fields relative to a query having a plurality of query terms. A field-weighted term frequency is computed for each query term based on field weights designated for individual fields in the document. A field-weighted document weight is computed for each query term based on the field-weighted term frequency for each field in the document. The field-weight score is computed based on the field-weighted document weights of the query terms.

In another implementation, a system for determining a field-weighted score for a document having multiple fields relative to a query having a plurality of query terms is provided. A field-weighted term frequency calculator computes a field-weighted term frequency for each query term based on field weights designated for individual fields in the document. A field-weighted document weight calculator computes a field-weighted document weight for each query term based on the field-weighted term frequency for each field in the document. A search engine computes the field-weighted score as a function of the field-weighted document weights of the query terms.

Other implementations are also described and recited herein.

Brief Descriptions of the Drawings

FIG. 1 illustrates an exemplary field-weighted search engine.

FIG. 2 illustrates an exemplary generation of a virtual document.

FIG. 3 illustrates an exemplary search engine employing a virtual document approach to provide a field-weighted search.

FIG. 4 illustrates an exemplary field weighting search engine employing a field weighting search algorithm to provide a field-weighted search.

FIG. 5 illustrates operations of an exemplary field-weighted searching process.

FIG. 6 illustrates operations of another exemplary field-weighted searching process.

FIG. 7 illustrates a system useful for implementing an embodiment of the present invention.

Detailed Description

FIG. 1 illustrates an exemplary field-weighted search engine 100. The search engine 100 receives a query 102 containing multiple query terms 104, 106, and 108. Each query term may include multiple component terms, such as if the query term is a phrase (e.g., the phrase "document management system" may be considered a single query term"). In addition, a query may include one or more operators, such as Boolean operators, constraints, etc., which are commonly supported by known search engines.

A plurality of documents, represented by documents 110, 112, 114, and 116, are available for searching. In practice, a search engine may search any number of documents and typically search collections containing large numbers

(e.g., millions) of documents. An indexing module (not shown) generates individual document statistics (e.g., 118, 120, 122, and 124) for each document. The document statistics are stored in an index 126.

The search engine 100 interrogates the index 126 to determine a search score 128 for each document based on the query 102 and the corresponding document statistics. Typically, the document scores 128 are then ranked in descending order to give the user a list of documents that are considered by the search algorithm to be most relevant to the query 102.

In the illustrated system, the search engine 100 represents a field-weighted search engine, which considers the structure of a document in its search algorithm. For examples, a simple document structure may include fields, such as title, abstract, and body. Other exemplary types of fields may include without limitation headings, sections, conclusions, and metadata fields.

A field-weighted search may be expected to be more accurate when the different fields are expected to include the same general type of language, although a less constrained selection of fields may be effective as well. For example, fields such as title, abstract, and body might be expected to share common and important terms that are indicative of the relevance of the document to a given search. In contrast, an author field tends to include names (a different "type" of language) that are not expected to be contained within the title, abstract, and body. Accordingly, one implementation omits an author field from a field-weighted search for this reason. Author fields and other "different types" of fields are, nevertheless, eligible for consideration in a field-weighted search, in any combination.

In one implementation, aspects of field weighting may be introduced by way of a field weighting indexer (e.g., the virtual document index generator of FIG. 3). In another implementation, fielding weighting features may be integrated into a field weighting search engine (e.g., the field weighting search engine of FIG. 4).

FIG. 2 illustrates an exemplary generation of a virtual document 200 from a document 202 that includes multiple fields: title field 204, abstract field 206, and body field 208. In this implementation, field weighting is introduced through the generation of a virtual documents 200 as influenced by field weight 210 (weight=5 in association with the title field 204), field weight 212 (weight=3 in association with the abstract field 206), and field weight 214 (weight=1 in association with the body field 208). A weight equaling zero may also be used, for example, to ignore the associated field in the search results.

In this implementation, each field of the document is replicated the number of times indicated by the field weight. The replicated field copies are concatenated to produce a field set (although other methods of combining the field copies into a field set may be employed). For example, the title 204 field is replicated five times to produce the title field set 216, the abstract field 206 is replicated three times to produce the abstract field set 218, and the body field 208 is replicated once to product an abstract fields set 220.

The three field sets are then concatenated together into the virtual document 200 (although other combinations are possible, such as mixing the field sets). As described with regard to FIG. 3, the virtual document may then be indexed to provide field-weighted virtual document statistics for the

document 202. These statistics may then be retrieved by a search engine to produce a field-weighted score for the document 202.

FIG. 3 illustrates an exemplary search engine 300 employing a virtual document approach to provide a field-weighted search. The search engine 300 receives a query 302 containing multiple query terms. A plurality 304 of documents is available for searching.

A virtual document index generator 306 inputs each document and a set of field weights 308. The index generator 306 generates a virtual document (not shown) from each input document, such as described with regard to FIG.2. The index generator 306 then generates virtual document statistics 310 for each virtual document. The virtual document statistics 310 may include term statistics 312 (including without limitation the frequencies of each term in the document (i.e., "term frequency" in the document) and the locations of each term in the document) and a document length table 314 (which indicates the length of the document).

To describe the operations for an exemplary field-weighted search using the virtual document approach, the following terms and notations are introduced. (Certain terms are preceded by a parenthetical indicating that the terms are "field-weighted". This description is meant to indicate that the document statistics of the virtual document implicitly include the influence of field weighting because the virtual document was constructed based on the field weights and the field lengths.):

Base Query Term Weight (w_i) – a weight applied to the query term i (e.g., the definite article "the" may be given less weight than other more informative query terms "structured" or "document")

(Field-weighted) Term Frequency ($tf_{i,d}$) - the number of occurrences of term i in virtual document d

(Field-weighted) Document Length (dl_d) - the length of virtual document d

(Field-weighted) document Weight (wd_i) – a function of a base query weight w for each query term i, the Term Frequency ($tf_{i,d}$) in the virtual document d, the document length (dl_d) of the virtual document d, and possibly other document-specific information (θ_d) (i.e., $wd_i(w_i,tf_{i,d},dl_d,\theta_d)$); this function may be linear or non-linear

Document Score (sc_d) – combines the document weights of all the terms of a query (indexed 1,..., V) into a single document score (e.g., $sc_d(wd_1,wd_2,...,wd_V)$); may be a linear combination or a non-linear combination

The search engine 300 then applies its search algorithm to generate a field-weighted document weight for each query term from the virtual document statistics. There exist a variety of methods for computing document weights and most can be used to compute the (field-weighted) document weight given the parameters w_i, $tf_{i,d}$, dl_d, *and* θ_d based on the virtual document. The search engine 300 then combines the field-weighted term weights for each document into a field-weighted document score 316. Typically, the document scores 316 for evaluated documents are then ranked in descending order to give the user a list of documents that are considered by the search algorithm to be most relevant to the query 302.

The virtual document approach described with regard to FIGs. 2 and 3 represents an effective implementation of a field-weighted search. Another implementation, discussed with regard to FIG. 4, employs document statistics for each document without resorting to generation of a virtual document. The

document statistics employed in the implementation illustrated in FIG. 4 includes field-specific information, such as the field location of each term (e.g., which field), field-specific term frequencies, and field lengths.

FIG. 4 illustrates an exemplary field weighting search engine 400 employing a field weighting search algorithm to provide a field-weighted search. To describe the operations for an exemplary field-weighted search, the following terms and notations are introduced:

Base Query Term Weight (w_i) – a weight applied to the query term i

Term Frequency ($tf_{i,d}$) - the number of occurrences of term i in document d

Field-specific Term Frequency ($tf_{i,d,f}$) - the number of occurrences of a term i in field f of document d

Document Length (dl_d) - the length of document d

Field Length ($dl_{d,f}$) - the length of field f in document d

Document Weight (wd_i) – a function of a base query weight w for each query term i, the Term Frequency ($tf_{i,d}$) in the document d, the document length (dl_d) of the document d, and possibly other document-specific information (θ_d) (i.e., $wd_i(w_i,tf_{i,d},dl_d,\theta_d)$)

Field-weighted Term Frequency ($ntf_{i,d}$) – a combination of field weights and field-specific term frequencies tf for a term i and a document d

Field-weighted Document Length (ndl_d) – a combination of field-weights and field lengths dl for a document d

Field-weighted Document Weight (fwd_i) - a function of a base query weight w for each query term, the Field-weighted Term Frequency ($ntf_{i,d}$) in the document d, the Field-weighted document length (ndl_d) of the document d, and possibly

other document-specific information (θ_d) (i.e., $fwd_i(w_i,ntf_{i,d},ndl_d,\theta_d)$); this function may be linear or non-linear

Field-weighted Document Score (fsc_d) – combines the weights of all the field-weighted document weights of all the terms of a query (indexed 1,..., V) into a single document score (i.e., $fsc_d(fwd_1,fwd_2,...,fwd_V)$); may be a linear combination or a non-linear combination

An index containing exemplary document statistics 402 are input to the field weighting search engine 400. The exemplary document statistics 402 include without limitation a term statistics 404 (including without limitation the frequencies of each term in each field (i.e., "field-specific term frequency") and the locations of each term in each field) and a document length table 406 (which indicates the length of each field). It should be understood that the document statistics 402 may include additional statistics, such as the locations of each term in the document, and the document length table 406 may include additional information, such the length of the overall document.

A multi-term query 408 is input to the field weighting search engine 400. A field-weighted term frequency calculator 410 inputs the query terms from the query 408 and field weights 412 The field-weighted term frequency calculator 410 also retrieves appropriate document statistics 402 (e.g., for each query terms). Based on these inputs, the field-weighted term frequency calculator 410 computes a field-weighted term frequency for each query term for each document. In one implementation, the field-weighted term frequency for a query term i and a document d is computed using the equation

$$ntf_{i,d} := \sum_{f \in document\ fields} m_f tf_{i,d,f} \qquad (1)$$

where m_f represents a field weight for field f and $tf_{i,d,f}$ represents a field-specific term frequency for the query term i, a field f, and the document d. However, it should be understood that the algorithm of Equation (1) is merely exemplary and that other algorithms may alternatively be employed.

A field-weighted document length calculator 414 inputs the field weights 412 and appropriate document statistics (e.g., field lengths). Based on these inputs, the field-weighted document length calculator 414 computes a field-weighted document length for each document. In one implementation, the field-weighted document length for a document d is computed using the equation

$$ndl_d := \sum_{f \in document\ fields} m_f dl_{d,f} \qquad (2)$$

where m_f represents a field weight for field f and $dl_{d,f}$ represents a field length for a field f and the document d. However, it should be understood that the algorithm of Equation (2) is merely exemplary and that other algorithms may alternatively be employed.

A field-weighted document weight calculator 416 computes a field-weighted document weight fwd_i for each term i in each document d as a function of the base query term weight, the field-weighted term frequency, the field-weighted document length, and possibly other document-specific information (e.g., $nfwd_i(w_i,ntf_{i,d},ndl_d,\theta_d)$). There exist a variety of methods for computing document weights and most can be used to compute the field-weighted document weight given these parameters. A document score calculator 418 computes a field-weighted document score 420 for each document searched. Typically, the document scores 420 for evaluated documents are then ranked in descending order

to give the user a list of documents that are considered by the search algorithm to be most relevant to the query 408.

FIG. 5 illustrates operations 500 of an exemplary field-weighted searching process using a virtual document approach. A field weight operation 502 determines the weights associated with each field of a set of searchable documents. Some documents may have different fields than other documents, and some fields existing in some documents in the set may not be weighted or may have weights set to zero.

A generating operation 504 generates field sets for each document based on the field weights and the identification of the fields of each document. Another generation operation 506 combine the field sets of each document to product an individual virtual document corresponding to each document. An indexing operation 508 analyzes each virtual document and generates virtual document statistics, which implicitly reflect the field-based influences of the document set.

An extraction operation 510 parses the query to determine the query terms. Base query term weights (e.g., w_i) are determined for each query term in a determination operation 512. A computing operation 514 computes for each virtual document the field-weighted document weights for each term from the field-weighted term frequency from the virtual document statistics. The computing operation 514 then computes a field-weighted document score based on the virtual document, which is associated with the original document. In some implementations, the document scores are ranked in ranking operation 516 and displayed in a user interface in the descending order of document score, reflecting each document's anticipated relevance.

FIG. 6 illustrates operations 600 of another exemplary field-weighted searching process. An extraction operation 602 parses the query to determine the query terms. Base query term weights (e.g., w_i) are determined for each query term in a determination operation 604.

A document index containing document statistics is retrieved by retrieval operation 606. Another retrieval operation 608 retrieves from the document index field-specific term frequencies and field lengths for each field of each document, based on the field weights. A calculation operation 610 computes the field-weighted term frequency and field-weighted document length for each document based on the field-specific term frequencies and field lengths. Another calculation operation 612 computes a field-weighted document weight for each term based on the field-weighted term frequency and field-weighted document length for each document. A computation operation 614 then computes a field-weighted document score based on the virtual document, which is associated with the original document. In some implementations, the document scores are ranked in ranking operation 616 and displayed in a user interface in the descending order of document score, reflecting each document's anticipated relevance.

Document weights w_i can also be modified by the number of fields in the document in which the term i occurs, represented by the multiple field factor $ff_{i,d}$. In one implementation, a suitable function may take the form:

$$nfwd_i\left(w_i, ntf_{i,d}, ndl_d, ff_i, \theta_d\right) := fwd_i\left(w_i, ntf_{i,d}, ndl_d, \theta_d\right) \cdot \frac{ff_i}{k_f + ff_i}$$

where k_f represents a scalar constant that controls the extent of the multiple field effect.

Document score fsc_d can also be modified based on detection of multiple query terms in a given field, represented by the multiple term factor bf_d. In one implementation, a suitable function may be implemented as follow:

$$bf_d := \frac{\max_{f \in \text{document fields}} \left\{ \sum_{i | tf_{i,d,f} \neq 0} fwd_i \right\}}{\sum_{i=1,\ldots,V} fwd_i}$$

Here bf_d represents a quantity between 0 and 1, which may be used to multiply the document score or may be used in a factor to modify the document score. In one implementation, the document score may be computed as:

$$nfsc_d := [k_m bf_d + (1 - k_m)] \, fsc(nfwd_1, nfwd_2, \ldots, nfwd_V)$$

One specific implementation is based on the BM25 ranking formula (see e.g., Robertson, S.E., Walker, S., Beaulieu, M.M., Gatford, M., Payne, A. (1995): Okapi at TREC-4, in NIST Special Publication 500-236: The Fourth Text Retrieval Conference (TREC-4): 73-96). In such a context, the field-weighted document weights fwd_i may be determined as follows:

$$fwd_i := \frac{(k_1 + 1) ntf_{i,d}}{k_1 \left((1 - b) + b \frac{ndl_d}{avndl} \right) + ntf_{i,d}} w_i$$

where ndl_d represents the field-weighted document length, $avndl$ is the average field-weighted document length across the collection of documents, and k_1 and b are free parameters. The basic query term weight w_i in the BM25 function is normally calculated as follows:

$$w_i = \log \frac{N - df_i + 0.5}{df_i + 0.5}$$

where df_i is the number of documents in which query term i occurs and N is the total number of documents in the collection. The field-weighted document score fsc_d may then be obtained by adding the field-weighted document weights:

$$fsc_d = \sum_i fwd_i$$

It should be understood that other scoring algorithms, either based on BM25 or otherwise, may also be employed.

Given the BM25 implementation described above, the parameters k_1 and b may be optimized once for the non-field-weighted configuration (e.g., all field weights equaling 1 represents an exemplary non-field-weighted configuration) for a given document collection and then be applied to all other combinations of fields weights for that collection. For example, assuming that k_1* and b* represent the optimal values for the non-field weighted case, k_1 and b may be optimized of a field-weighted case by:

(1) Calculating the average term frequency over all terms and all documents in the non-field-weighted configuration, $atf_{unweighted}$.

(2) For a particular combination of field weights, calculating the average term frequency, $atf_{weighted}$ (e.g., an average of the field-weighted term frequencies of all terms and all documents in the field-weighted configuration).

(3) Calculating the optimal b for the field weight combination as $b=b$*.

(4) Calculating the optimal k_1 for the field weight combination as

$$k_1 = k_1^* \frac{atf_{weighted}}{atf_{unweighted}}.$$

The exemplary hardware and operating environment of FIG. 7 for implementing the invention includes a general purpose computing device in the form of a computer 20, including a processing unit 21, a system memory 22, and a

system bus 23 that operatively couples various system components include the system memory to the processing unit 21. There may be only one or there may be more than one processing unit 21, such that the processor of computer 20 comprises a single central-processing unit (CPU), or a plurality of processing units, commonly referred to as a parallel processing environment. The computer 20 may be a conventional computer, a distributed computer, or any other type of computer; the invention is not so limited.

The system bus 23 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, a switched fabric, point-to-point connections, and a local bus using any of a variety of bus architectures. The system memory may also be referred to as simply the memory, and includes read only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system (BIOS) 26, containing the basic routines that help to transfer information between elements within the computer 20, such as during start-up, is stored in ROM 24. The computer 20 further includes a hard disk drive 27 for reading from and writing to a hard disk, not shown, a magnetic disk drive 28 for reading from or writing to a removable magnetic disk 29, and an optical disk drive 30 for reading from or writing to a removable optical disk 31 such as a CD ROM or other optical media.

The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical disk drive interface 34, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer-readable instructions, data structures, program modules and other data for the computer 20. It should be appreciated by those skilled in the art that

any type of computer-readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, random access memories (RAMs), read only memories (ROMs), and the like, may be used in the exemplary operating environment.

A number of program modules may be stored on the hard disk, magnetic disk 29, optical disk 31, ROM 24, or RAM 25, including an operating system 35, one or more application programs 36, other program modules 37, and program data 38. A user may enter commands and information into the personal computer 20 through input devices such as a keyboard 40 and pointing device 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port, or a universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via an interface, such as a video adapter 48. In addition to the monitor, computers typically include other peripheral output devices (not shown), such as speakers and printers.

The computer 20 may operate in a networked environment using logical connections to one or more remote computers, such as remote computer 49. These logical connections are achieved by a communication device coupled to or a part of the computer 20; the invention is not limited to a particular type of communications device. The remote computer 49 may be another computer, a server, a router, a network PC, a client, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 20, although only a memory storage device 50 has been illustrated

in FIG. 7. The logical connections depicted in FIG. 7 include a local-area network (LAN) 51 and a wide-area network (WAN) 52. Such networking environments are commonplace in office networks, enterprise-wide computer networks, intranets and the Internet, which are all types of networks.

When used in a LAN-networking environment, the computer 20 is connected to the local network 51 through a network interface or adapter 53, which is one type of communications device. When used in a WAN-networking environment, the computer 20 typically includes a modem 54, a network adapter, a type of communications device, or any other type of communications device for establishing communications over the wide area network 52. The modem 54, which may be internal or external, is connected to the system bus 23 via the serial port interface 46. In a networked environment, program modules depicted relative to the personal computer 20, or portions thereof, may be stored in the remote memory storage device. It is appreciated that the network connections shown are exemplary and other means of and communications devices for establishing a communications link between the computers may be used.

In an exemplary implementation, a search engine, a virtual document index generator, a field-weighted term frequency calculator, a field-weighted document length calculator, and other modules may be incorporated as part of the operating system 35, application programs 36, or other program modules 37. Document statistics, search scores, and other data may be stored as program data 38.

The embodiments of the invention described herein are implemented as logical steps in one or more computer systems. The logical operations of the present invention are implemented (1) as a sequence of processor-implemented steps executing in one or more computer systems and (2) as interconnected

machine modules within one or more computer systems. The implementation is a matter of choice, dependent on the performance requirements of the computer system implementing the invention. Accordingly, the logical operations making up the embodiments of the invention described herein are referred to variously as operations, steps, objects, or modules.

The above specification, examples and data provide a complete description of the structure and use of exemplary embodiments of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.